Getting Started with Full Stack Development: A Project Idea

2

LEARNING OBJECTIVES

After completing this chapter, you will be able to understand:

- E-commerce and its advantages.
- Entity Relationship Diagram (ERD) and an entity.
- How to create a flowchart from the development perspective.
- Planning and execution of a development project.
- The designing of front-end structure and interfaces.
- The designing of web services APIs and the definition of required endpoints.
- GET and POST methods to send data from client to server.

2.1 | Introduction

This book is intended to be a practical guide to full stack development. It not only covers the theoretical part but also focuses on the practical use. Keeping this objective in mind, we will be learning through an example application. As we learn about the technologies, we will apply this newly gained knowledge to design a real-life application. In this chapter, we will introduce a project idea and start building it. For this exercise, we will use an e-commerce example. For those who are new to the concept of e-commerce, we will explore this in detail as well.

2.2 | Project Outline

This project has been specially designed to take you through the full development process of front-end web development and back-end scalable web service endpoints. We will be considering all the major functionalities of an e-commerce website such as a shopping cart, customer relationship management (CRM) integration, payment gateway, etc. We will think of all the design choices from the user's point of view and consider security and performance.

Before we begin with the project planning and technical diagrams, let us first understand the e-commerce domain.

2.3 | What is E-Commerce?

E-commerce is the process of buying and selling of goods through electronic means (Internet or apps). Over the last decade, the popularity of e-commerce has hugely increased and it is replacing traditional brick and mortar stores. E-commerce helps us buy products on a global scale anytime during the day or night.

2.3.1 Advantages of E-Commerce

Following are the main advantages of e-commerce:

1. **Low start-up cost:** Physical stores have to spend tons of money to create an establishment and buy inventory, sales equipment, and more. Plus, having a physical store means that you have to pay your staff to work. On the other hand, starting up an e-commerce is quite cost-effective. You just need to spend on the website or app development, a tie-up with a postal service, and inventory. This naturally helps the entrepreneurs to keep the costs down.

2. **24/7 chances of income:** When you open a traditional store, you have to decide particular opening and closing hours. You cannot simply keep your store open all the time as it will need you to spend more resources on your store. Having an e-commerce platform means you are always ready for business. The web is already there for customers to navigate to your website. Thus, you do not need to have employees to be working in the night to process the orders.
3. **Sell anywhere:** Compared to traditional stores, you can sell products anywhere depending on the reach of the courier service. However, having a shop means you will have only one location until and unless you open other branches (which is expensive).
4. **Get access to customer data:** Building a relationship with the customer is very important to boost sales. In traditional stores, the customers are often uncomfortable to give away their personal details like phone number, address, e-mail, etc. Online e-commerce makes this simple by creating user accounts for the customers where they need to put their personal details to process orders. You can even conduct marketing surveys for products in your inventory with the help of e-commerce.
5. **Helps process a large number of orders at once:** Surely you can manage a large number of orders in a traditional shop. However, you will need a large workforce and space to accommodate such power. E-commerce makes it so simple and automated that you really do not need to do much. In e-commerce, the customers can easily place single or multiple orders according to their own schedule.
6. **Easier to encourage buying:** What looks the best sells the best. Obviously, you can decorate your traditional store with decorations and place your products smartly. However, you will be spending a lot on just creating a good presentation. On the other hand, in e-commerce, everything is much easier by creating a vivid, fluid, and intuitive website design.

2.3.2 Important Considerations while Developing an E-Commerce Website

There are quite a few factors to take into account while developing an e-commerce website. Some of the important ones are discussed below.

1. **Knowing target customer base:** Before we begin designing an e-commerce website, it is important to understand the target market. This will tell us how large a market we are serving so that we can design our application accordingly. This information helps in selecting the technologies we will need. For example, for a larger customer base like Amazon.com, our traditional RDBMS database will not give good results as Amazon sells millions of products in a short amount of time. For this, we need to consider NoSQL databases like Cassandra which are faster than RDBMS databases. Also, knowing our customer base will tell us what other features we need to consider. For example, for a younger target market, we need to accommodate social media marketing module where the backend system will generate social media posts, track user actions, build personalized liked categories products, etc. This type of analytics is needed to make the web application more user-focused. We will also need to consider adding algorithms to help users in buying and product selections by presenting relevant products, offers, services, etc.
2. **Choosing the right technologies:** Once we know our target market, it is important to select the best possible technology stack for the development. We have already discussed the RDBMS and NoSQL databases and their comparison; there are other areas which are important in terms of technology selection. For example, product selection, offers, and promotions will be different for mobile users than web users. Hence, in a younger target market case, we need to select technologies which support mobile front-end like push notifications, real-time tracking, location-based product selection, etc. We also need to consider web servers we will be using for production deployment. Every server out there has pros and cons, so we need to weigh the selection criteria based on our target market use. The strength and weaknesses of each platform are quite different from each other. Also, our development framework might change according to server selection like configuration files, libraries, etc. Hence, a detailed analysis of these web server platforms is necessary prior to development phase. This will lower the scope for mistakes and errors. Here are a few things we should consider while selecting a technology stack:
   - CRM, ERP, database integration and compatibility.
   - Scope for e-commerce website customization.
   - Payment gateway integration.
   - Robust back-end support.
   - Mobile friendly and responsive e-commerce.
   - E-commerce marketing.

3. **Design elements of the application:** It is important to understand that there are various other elements that are equally important as look and feel. While designing an e-commerce application, it is extremely vital to consider performance, security, accessibility, scalability, etc. of the platform. Here are a few other things we should consider while designing the application:
   - **Speed:** The users do not like to wait for long while websites get load. Search engines like Google further lower down this. Thus, speed is power.
   - **Uptime:** We must focus on the uptime records of our application. Having a great uptime is essential for an e-commerce application as it is the main purpose of having an online shop, so that customers can shop any time from any place. This is an important step while considering a web server to make sure it is compatible with our technology stack.
   - **Support:** We must consider customer support module while designing the application. People will not use the platform if they are not able to communicate with the support associates for any queries or help. Many users may also need support in product selection, buying choices, etc.
   - **Scalability:** E-commerce applications can see seasonal surges in traffic and hence it is important to make sure that the servers are capable of handling a large amount of traffic. We must consider this while writing code as many performance problems occur due to bad code.
   - **Security:** This is another vital area to consider as people will not like to get their credit card information in the wrong hands or leak their personal information. Having a secure application is crucial to building trust among the customers. There are various steps we could take while writing code, configuring servers, communicating with third-party applications like CRM system, Payment Gateways, etc. So focusing on security is essential.
4. **Thorough testing:** Testing should be an essential part of the development. We should not wait until the completion of the development; rather we should consider testing our code in units. This is where unit testing comes handy. We must consider writing unit tests for our code so that we find problems as early as possible in the development. Unit tests can also expose any security or performance leak in the application. Once the application is complete with successful unit tests, we still cannot directly make it public without taking it through a quality check process. Intense testing should be done in order to check usability, customer convenience, bugs, and ultimately a good shopping experience. The testing can be carried out in various web browsers, platforms, and multiple devices. This testing could be manual or automated. It can even be the combination of the two forms. We must examine for slow loading of webpages, poor navigation, broken links, and bugs to provide a seamless experience to the user. Our aim should be to catch the bugs and flaws as early as possible in the development phase. If these bugs pass the development phase they would become very expensive to manage. So what exactly should be tested during the testing phase?
   - **User flow of the website:** We must check each and every page accessible to the users. Starting from the home page to the menu, product listing, search bar, and results. Give the best attention to the payment and checkout window.
   - **Functionalities of the website:** This includes the categories, the pages, user registrations, filters, shopping cart, payment options, shipping options, and all other important features of the website.
   - **Security of the website:** Security is vital in an e-commerce website. This applies not just to the payment page where the customers enter their sensitive information such as credit card details, but also to every page of the website. Having a secure website builds trusts among users.
   - **Testing the compatibility of the website:** Everyone has different devices and different browsers while browsing the web. Thus, we must test the website on different web browsers, different platforms, and multiple devices such as mobile/tablets/laptops.
   - **Performance and discoverability of the website:** No one waits for ages to open a website. Thus, it is very important to monitor the loading time of your website. Try to catch the broken links, check the SEO ranking in different search engines.

2.4 | Required Entities

Following are some of the essential entities we will need to consider in developing the end-to-end e-commerce application. We also need to identify the relationships between these entities to make sure we consider them properly in the design.

1. **Customer:** This is the main entity in the e-commerce application. Customer is the one who buys products from the e-commerce shopping portal.
2. **Address:** This entity is related to the customer entity to store multiple addresses for a customer.

3. **Currency:** This entity is to define the preferred currency for the customer.
4. **Product:** This entity represents physical or digital goods to be sold on this e-commerce platform.
5. **Shopping cart:** This entity defines a place where the customer stores his/her chosen products.
6. **Coupon:** This entity defines discounts on products.
7. **Offer:** This entity specifies promotional offers for products.
8. **Vendor:** This entity sells products on the e-commerce platform.
9. **Order:** This entity defines a consolidated place for chosen products to be purchased by a customer.
10. **Shipping:** This entity defines the product shipment process for customers.
11. **Payment:** This entity is for financial transactions that took place in a purchase process.
12. **Invoice:** This entity is a bill generated for an order.
13. **Inventory:** This entity defines a place to store product quantity-related information.
14. **Catalog:** This entity defines a ledger that shows products, offers, and discounts related details.
15. **Warehouse:** This entity provides a physical storage place for products.

QUICK CHALLENGE Extend the functionality of this e-commerce platform and add more entities. Think of various different possibilities to enhance the user experience.

2.5 | Entity Relationship Diagram

Now let us consider these entities and figure out relationships between them. In Figure 2.1, we will try to link the entities to understand the linkage between them and learn how to handle them.

QUICK CHALLENGE Create new entity relationship diagram (ERD) for the additional entities you have figured out in the earlier exercise.

2.6 | UML Class Diagram

Based on the entity relationship diagram (ERD), we can create a class diagram that will show the table relationships as per entities. Figure 2.2 also contains fields of the classes and their types.

The following explains how to read the relationships on the entities:

1. Single dash on both the entities shows one-to-one relationship between those entities.
2. Single dash on one and three dashes on the other entity show one-to-many relationship.
3. Three dashes on one and single dash on the other entity show many-to-one relationship.
4. Three dashes on one and three dashes on the other entity show many-to-many relationship.

QUICK CHALLENGE Extend this class diagram with the new entities you have identified in the earlier exercise.

2.7 | Flowchart

Flowchart is an important element in designing an application because it makes us think of all the possibilities in the user flow. It shows each and every step of the application and how data and decisions are flowing within modules. Before we take a look at various flowcharts, let us first understand the elements and their meanings. Following are a few elements of a flowchart.

1. **Oval:** It defines start and end of the flowchart and marks it accordingly. Use start to begin the flowchart and end to complete the flowchart.

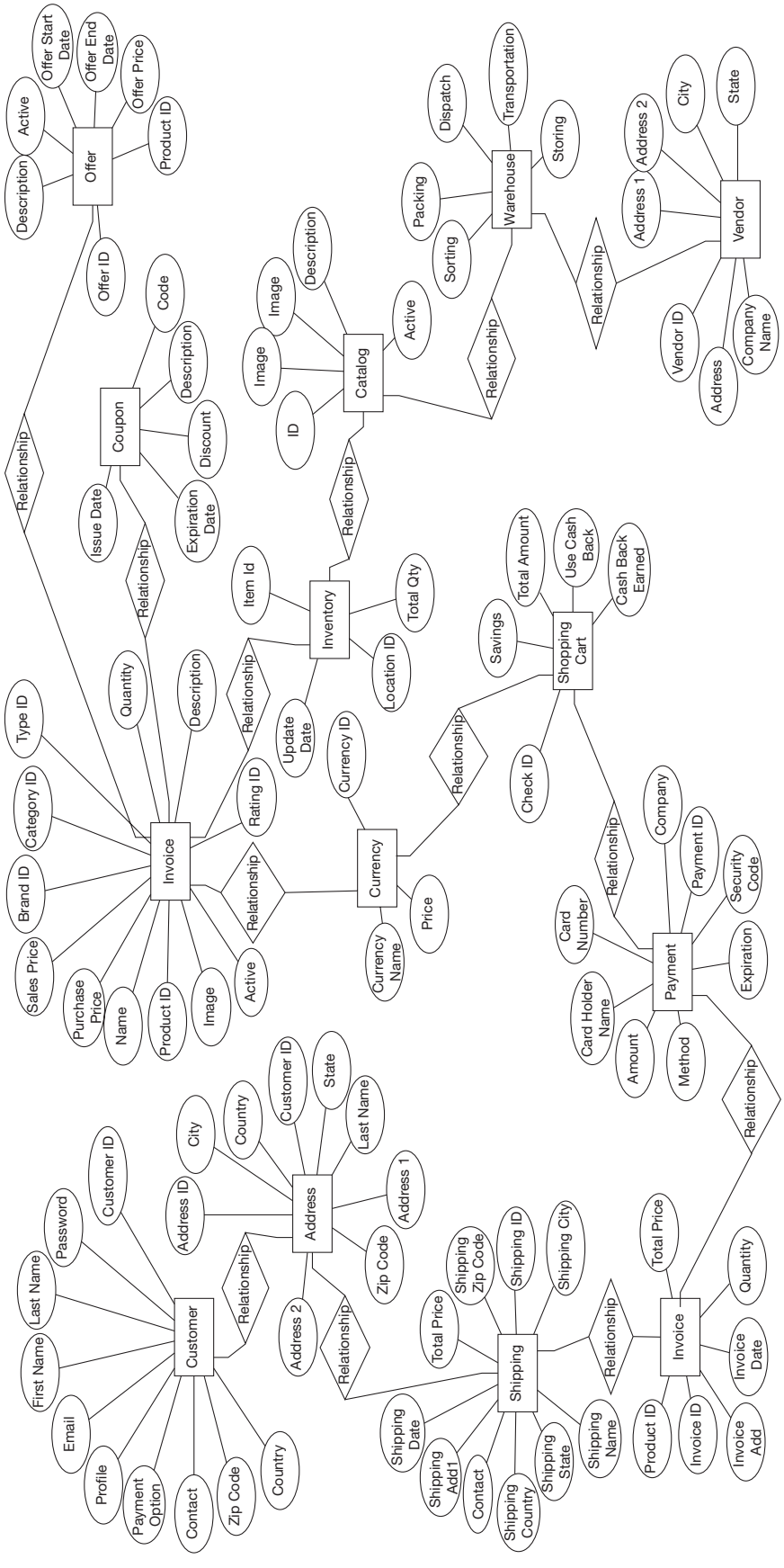


Figure 2.1 Entity relationship diagram of e-commerce entities.

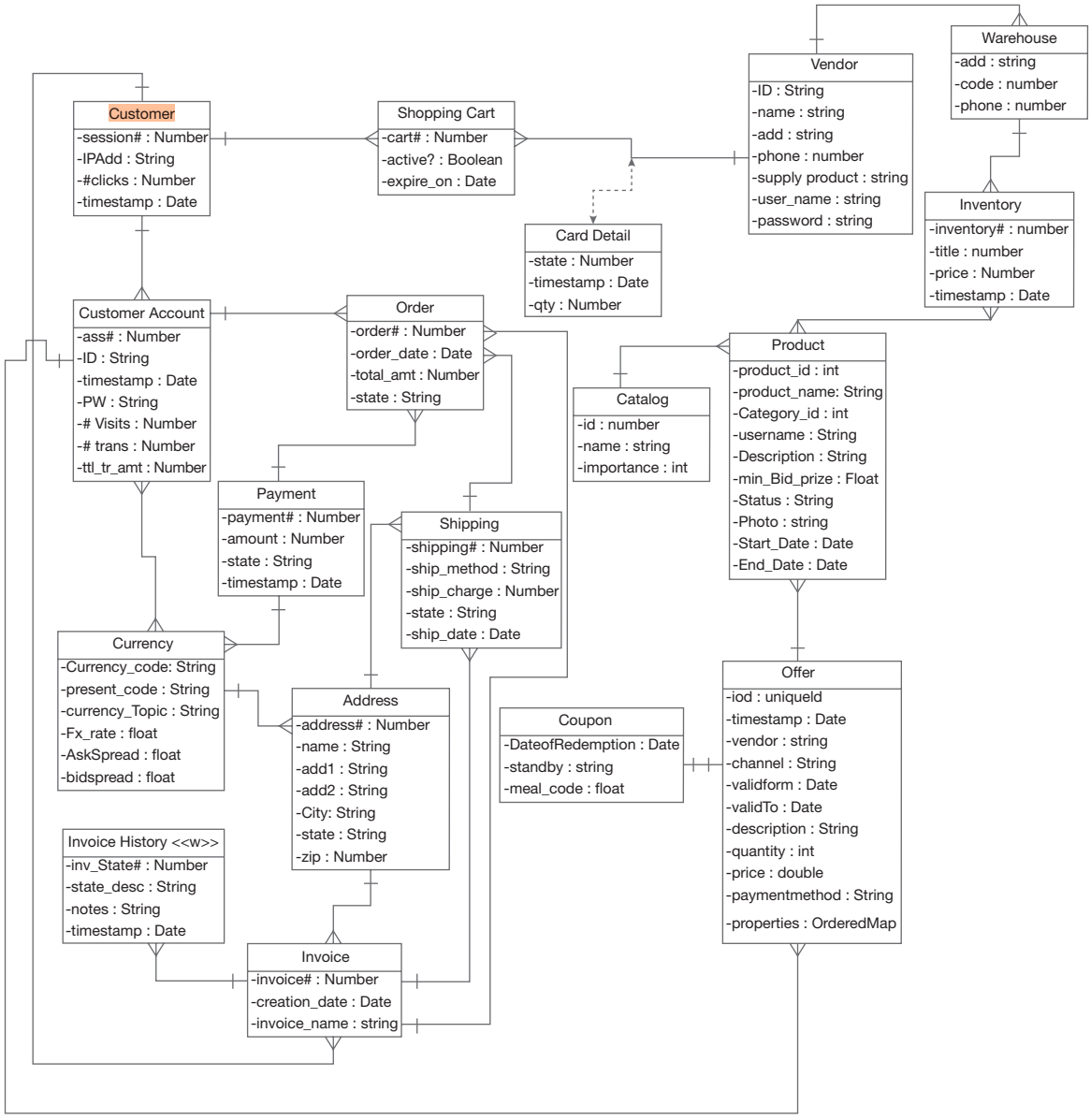
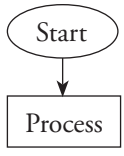


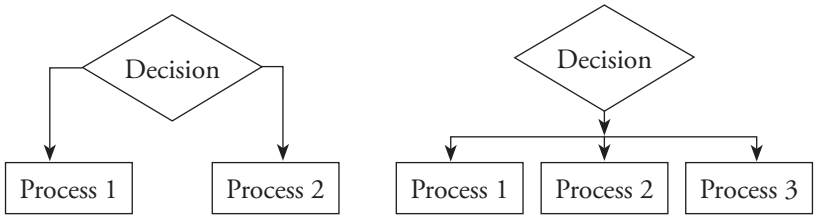
Figure 2.2 UML class diagram of e-commerce classes.

2. **Rectangle:** It is used to define a process step in the flowchart. It is a simple rectangle which contains the name of the process.

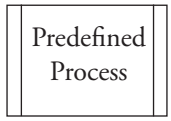
3. **Arrow:** It is used to define a direction of the processing flow. Use this to show how the process data flows in the flowchart.



4. **Diamond:** It is used to specify the decisions to be made in the flow. With this you can show multiple choices as well.



5. **Rectangle with three sections:** It is used to specify predefined process, which is also called *subroutine*. This process is formally defined somewhere else and mainly used for expressing a subprocess.



2.7.1 Login–Registration–Ordering Flowchart

The flowchart in Figure 2.3 shows how the user can login to the front-end website and proceed on completing an online order. It also shows a process where a user can register if not already registered.

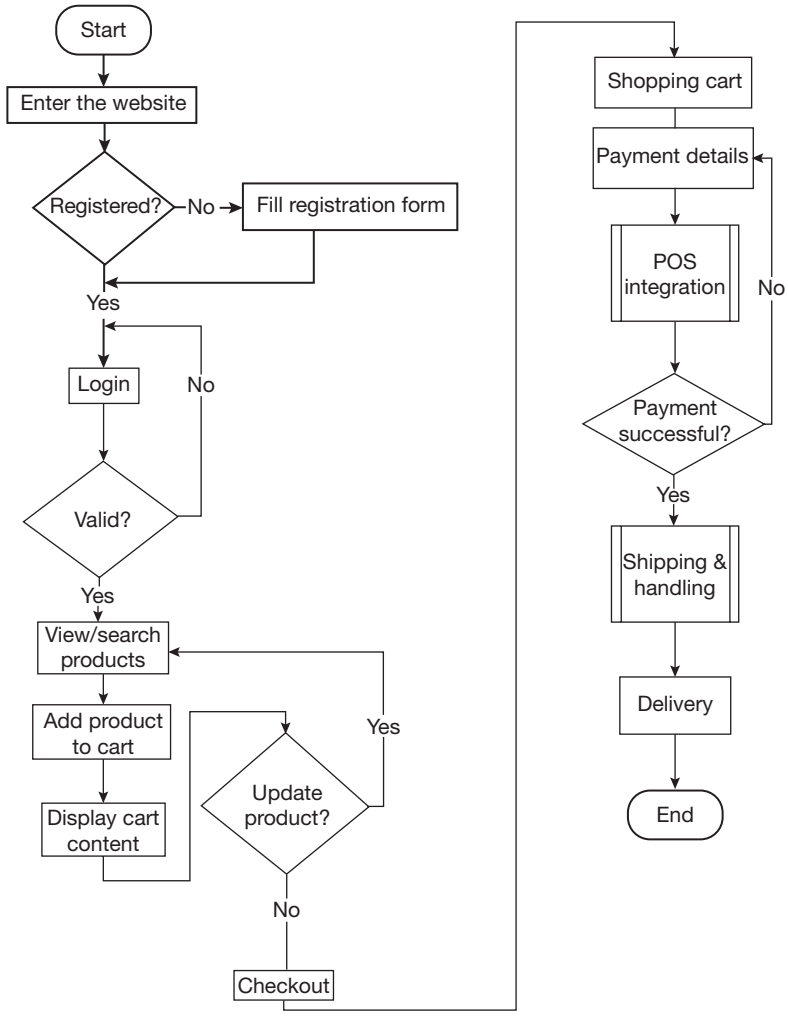


Figure 2.3 Login–registration–ordering flowchart.

Login Registration Process Flow:

Step 1: To begin shopping, the customer will need to type the URL or name of the shopping website into a web browser to enter into the website.

Step 2: Now, the customer will need a registered account on the website to start shopping. If he/she does not have an account then the website will prompt the customer to create one. The diamond box checks this condition. The website will ask for a valid address, mobile number, name, etc.

Step 3: Once the customer has created an account on the website, they need to use their login credentials to proceed further.

Step 4: After logging into the website, the customer can start shopping.

Step 5: The customer can view different pages of the site or use search options to browse for products to shop.

Step 6: Once the customer has selected a product after reading its details and features, they can add it to their cart for further billing.

Step 7: The customer needs to click on the show cart items to display cart content.

Step 8: Now, the customer needs to decide whether he/she wants to make modifications to the products on his/her cart. For example, changing the product quantity. The diamond box checks this condition.

Step 8.1: If the customer has decided to update the product quantity, then he/she can change it by using scrolls. However, if they do not want to update anything, then he/she will proceed directly to the next step, that is, checkout.

Step 9: In the checkout window, the customer will be provided with the final summary and pricing of the product he/she is purchasing. If the customer is still not satisfied with the selected products, he/she can go back to Step 4 and repeat the process.

Step 10: Next, the customer is directed to the payment options.

Step 11: The customer will need to select his/her mode of payment to pay for the products. There will be multiple options for payment.

Step 12: The website will guide the customer through the POS integrated checkout system. POS integration helps e-commerce websites to effectively manage sales and purchases from scanning product information and generating receipts after the use of credit/debit card.

Step 13: Now, it is up to the bank and vendor to verify and authorize the payment. If it is completed, then it will show a success message and the next step will begin. However, if the payment fails, the customer will be redirected to the payment details page to try again or choose other payment options. The diamond box checks this condition.

Step 14: If the payment is successful, the process of shipping and handling begins in which the vendor and a logistic company plays the important role of handling and delivering the product(s) at the doorstep of the customer.

Step 15: The product will be delivered to the customer within an estimated delivery time.

2.7.2 Cart Finalization Flowchart

The flowchart in Figure 2.4 shows how a customer can complete ordering the desired products. It takes us through the order finalization stage where the customer can view and update the order if required.

Cart Finalization Process Flow:

Step 1: First the website will get information about the cart order.

Step 2: Then from the cart order the cart item information will be gathered.

Step 3: Then it will redirect the page to the order info page for further processes.

Step 3.1: If the customer wants to further update the cart, he/she can still finalize it before posting.

Step 3.2: Once the cart items have been finalized by the customer, he/she post cart item(s).

Step 4.1: The system calculates the shipping rate for the selected products in the cart and adds it to the final bill.

Step 4.2: After finalizing the carts items, the customer can also browse the misc menu option for tiny modifications in the order. For example, the customer can gift wrap the product or schedule the delivery date according to his preference.

Step 5: Once Steps 4.1 and 4.2 is complete, the cart items are ready to be posted for billing process.

Step 6: The system checks if the entered information is valid or not to complete the next process. To gather the information, the system repeats the Step 2 (get cart item info). However, if the information is not valid, then the process terminates.

Step 7: The system calculates the final amount to be paid by the customer.

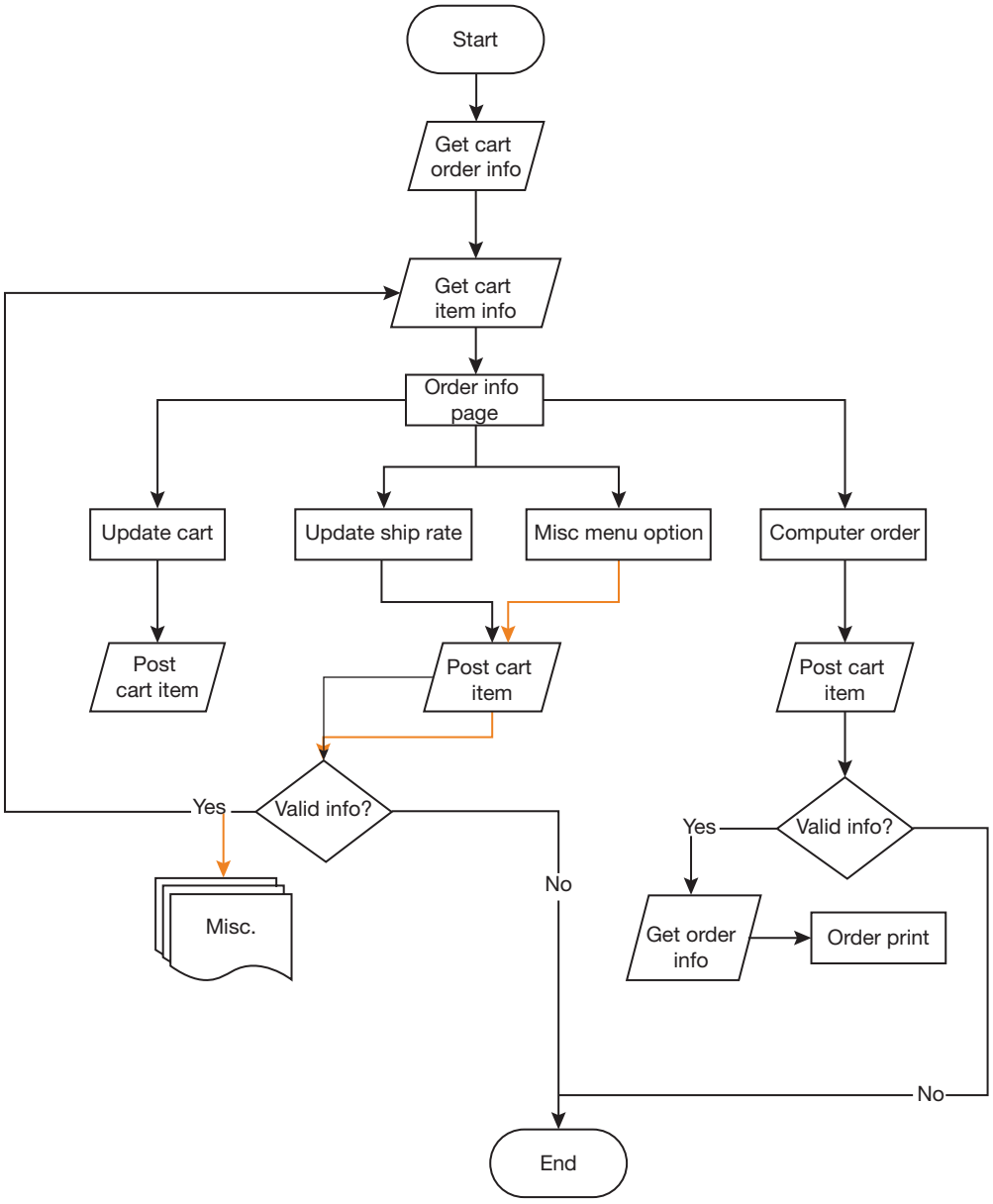


Figure 2.4 Cart finalization flowchart.

Step 8: The items in the cart gets posted to create an invoice.
Step 9: The system verifies if the information to be printed is valid or not. If the information is not valid, then the process gets terminated.
Step 10: If the information is correct, then system gathers order info. This may include the specifications of the product.

2.7.3 Order Processing Flowchart

Figure 2.5 shows the order processing flow. This goes through the back-end process to fulfill the order placed by a customer.

Order Processing Flow:

Step 1: First, the customer will place an order on the e-commerce website and the admin will receive a notification.
Step 2: Now, the admin will check the details of the order placed by the customer.

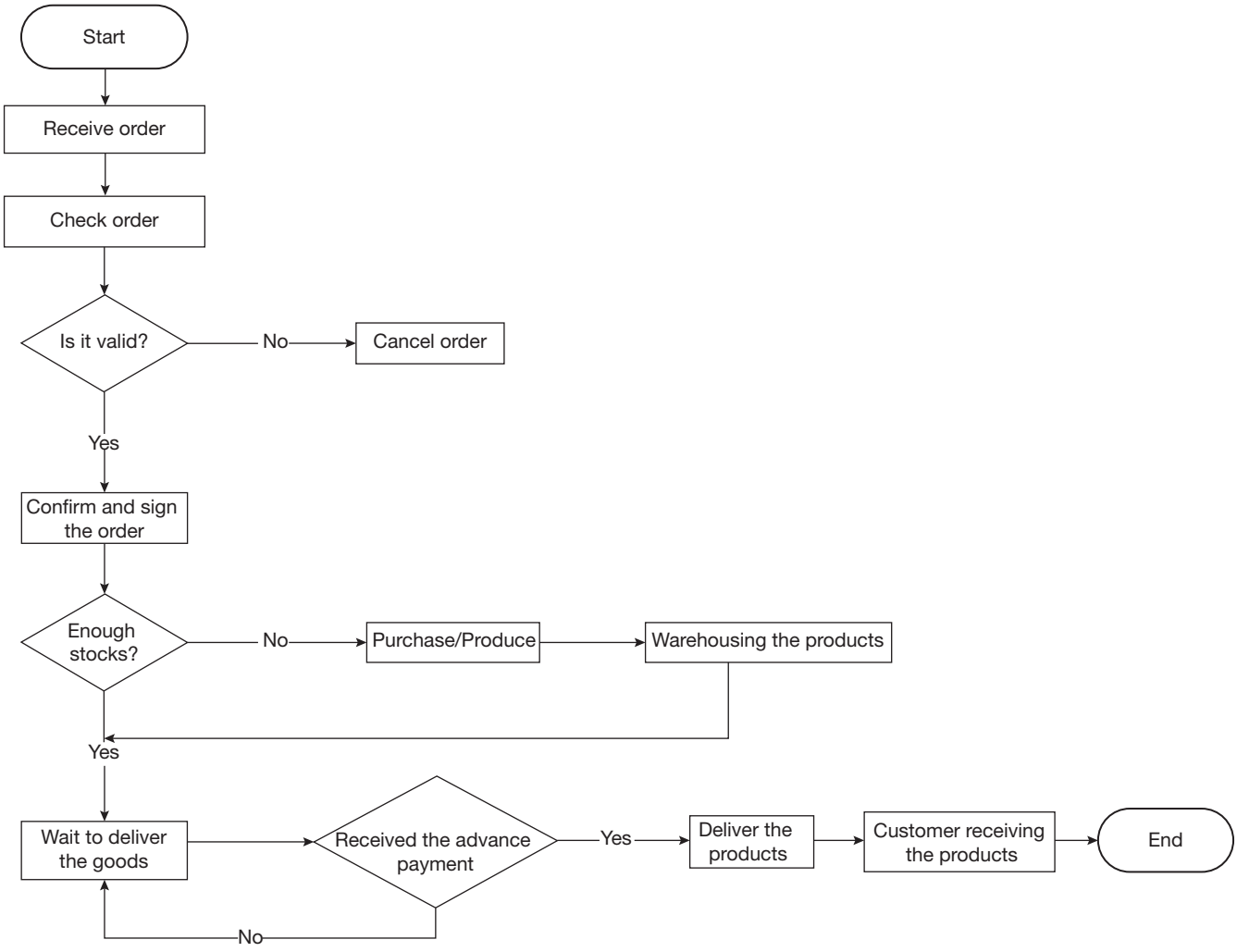


Figure 2.5 Order processing flowchart.

Step 3: The admin will determine whether the order is valid or not. For example, e-commerce websites do not let users to stockpile products. In such cases, the order will be cancelled.
Step 4: If the order seems to be valid, then the admin confirms and sign the order for further processes.
Step 5: Now, the admin needs to check on the stocks to fulfill the order.
Step 5.1: If there is insufficient stock in the inventory, then the admin will need to purchase or produce the products from other vendors.
Step 5.2: Once the products have been purchased or produced, they need to be added to the warehouse of the e-commerce owner.
Step 6: Now, the admin needs to wait to deliver the products before he/she receives payment.
Step 7: Advance payment needs to be received before beginning the delivery the goods. However, if the payment is not received, then the admin will need to wait to dispatch the products.
Step 8: The product(s) will be dispatched for delivery once the advance payment has been received.
Step 9: At the end, the customer will receive the product within an estimated delivery time.

2.7.4 Vendor Order Processing Flowchart

Figure 2.6 shows how vendor processes the order once received from a customer. It goes through the stock verification process and ends with shipment and payment collection in case of cash on delivery (COD).

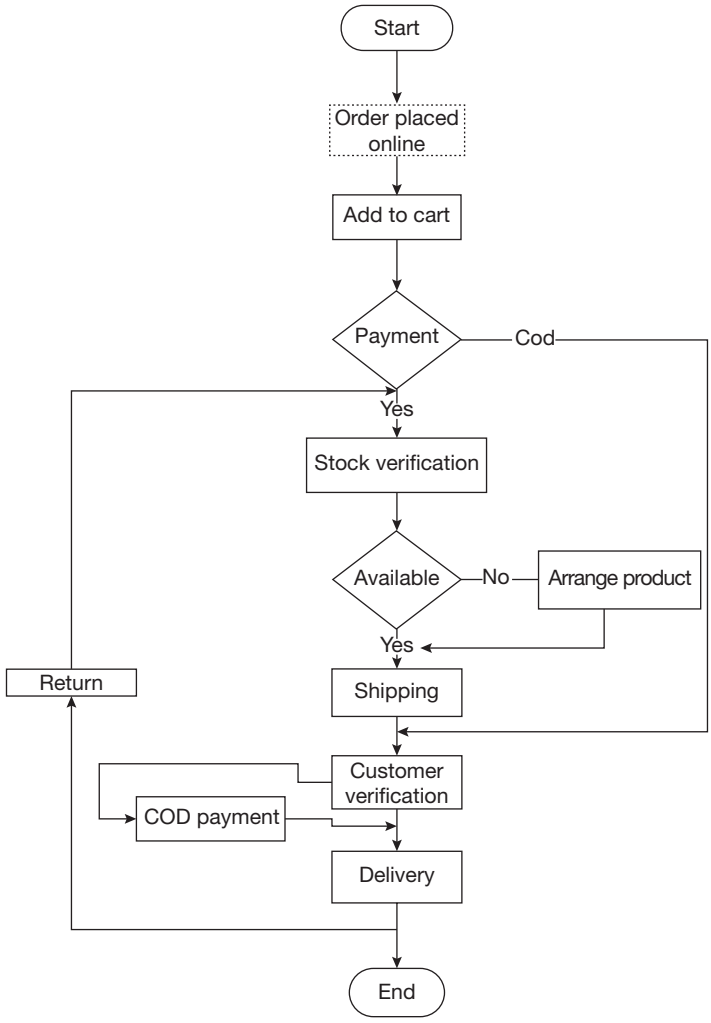


Figure 2.6 Vendor order processing flowchart.

2.7.5 Admin Dashboard Flowchart

Figure 2.7 shows the admin dashboard flow where an admin user can login to the back-end system and perform specific tasks.

Admin Dashboard Process Flow:

Step 1: To access the admin dashboard, the admin will need to visit the login page for admins.
Step 2: Here the admin is needed to enter a valid username and password.
Step 3: If the entered user name and password are correct, then it will direct the admin to the admin dashboard. However, it will redirect the admin to the login page if they have entered the wrong credentials.
Step 4: Once the admin has logged in, they will have access to B2B, account manager, and sales report.
Step 4.1: In the B2B user tab, admins can manage their connections with other vendors.
Step 4.2: Admins can also view their detailed customer information to get informed.
Step 4.3: The product catalog will help the admin to manage products on their website. They can add/edit/remove products from there.
Step 5: Account Manager Tab lets the admin edit important details. They can tweak website settings from there.
Step 5.1: Admins can also browse customer information to tweak the store according to the nature of the customer to improve sales. This will show a list of registered users on their website.

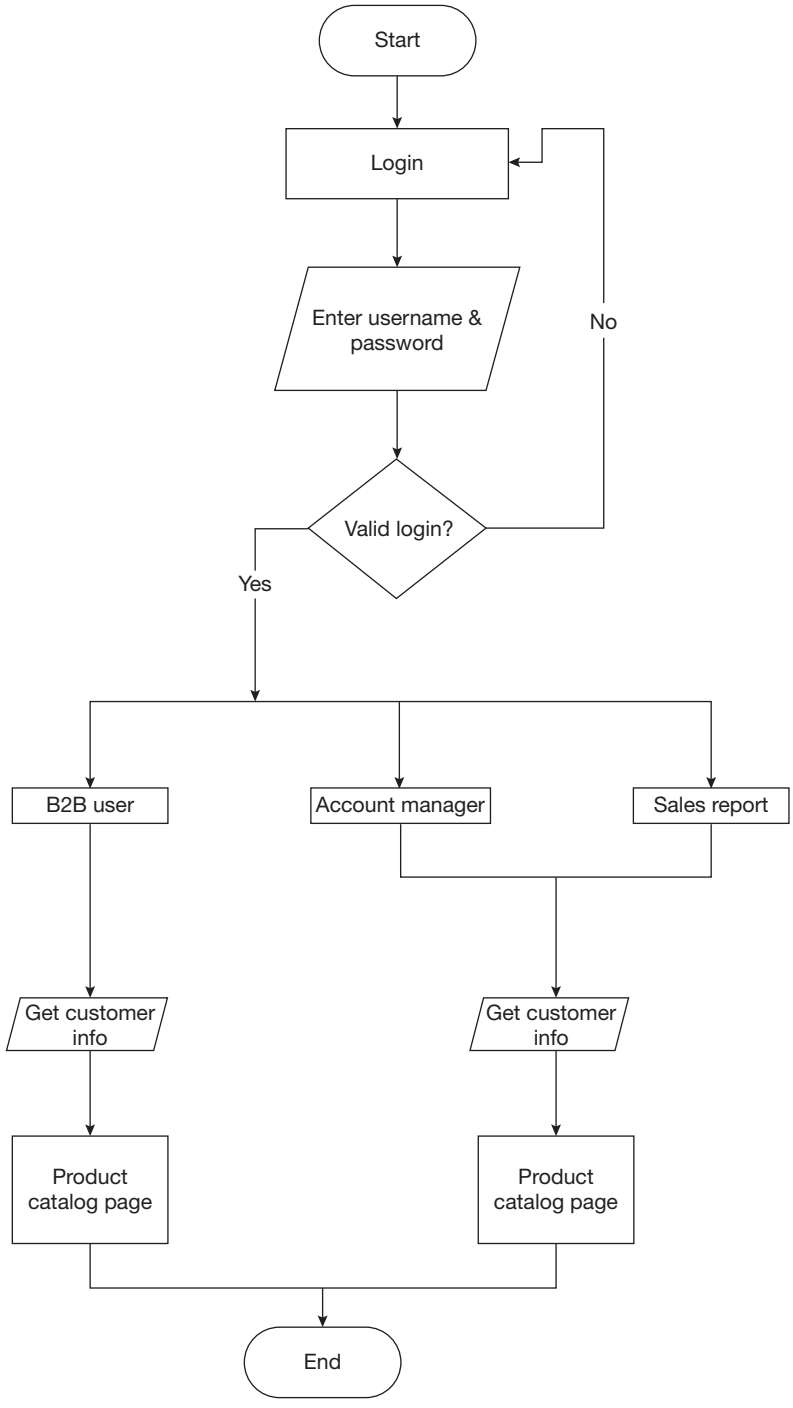


Figure 2.7 Admin dashboard flowchart.

Step 5.2: Admins can then browse the product catalog page to tweak the products according to their customer's nature.
Step 6: The Sales Report Tab will show a comprehensive report of sales of their products.
Step 6.1: The admins can get the customer information through the sales report tab and know their preference and reach.
Step 6.2: Then the admins can follow Step 5.2 to tweak the store.

QUICK CHALLENGE

As we have seen in the above examples, flowchart can be made for each major activity in the system. So far, we have covered login, registration, order processing, cart finalization, vendor order processing, and admin dashboard.

Create flowcharts for remaining activities – payment processing, refund and return management, user my account page, user add to cart flow, admin add product process, admin add offer and discount management processes, user delivery tracking process, and admin financials page to show sales. Also, think of adding third-party integration in the flowchart like external chat system, other website integration, Google analytics, affiliate programs, etc. Following are the steps you could take to complete this challenge:

- Identify major activity in the system.
- Identify all the entities related to that activity.
- Figure out the entire flow to complete that activity.
- Consider the conditions and put them in the diamond shape boxes.

2.8 | Front-End Page Flow Design

The front-end page flow diagram in Figure 2.8 shows how the pages communicate with each other and how users interact with the pages. It also shows process flow and the conditions attached to the flow. The flowchart is a login, registration, search, purchase, and payment flowchart.

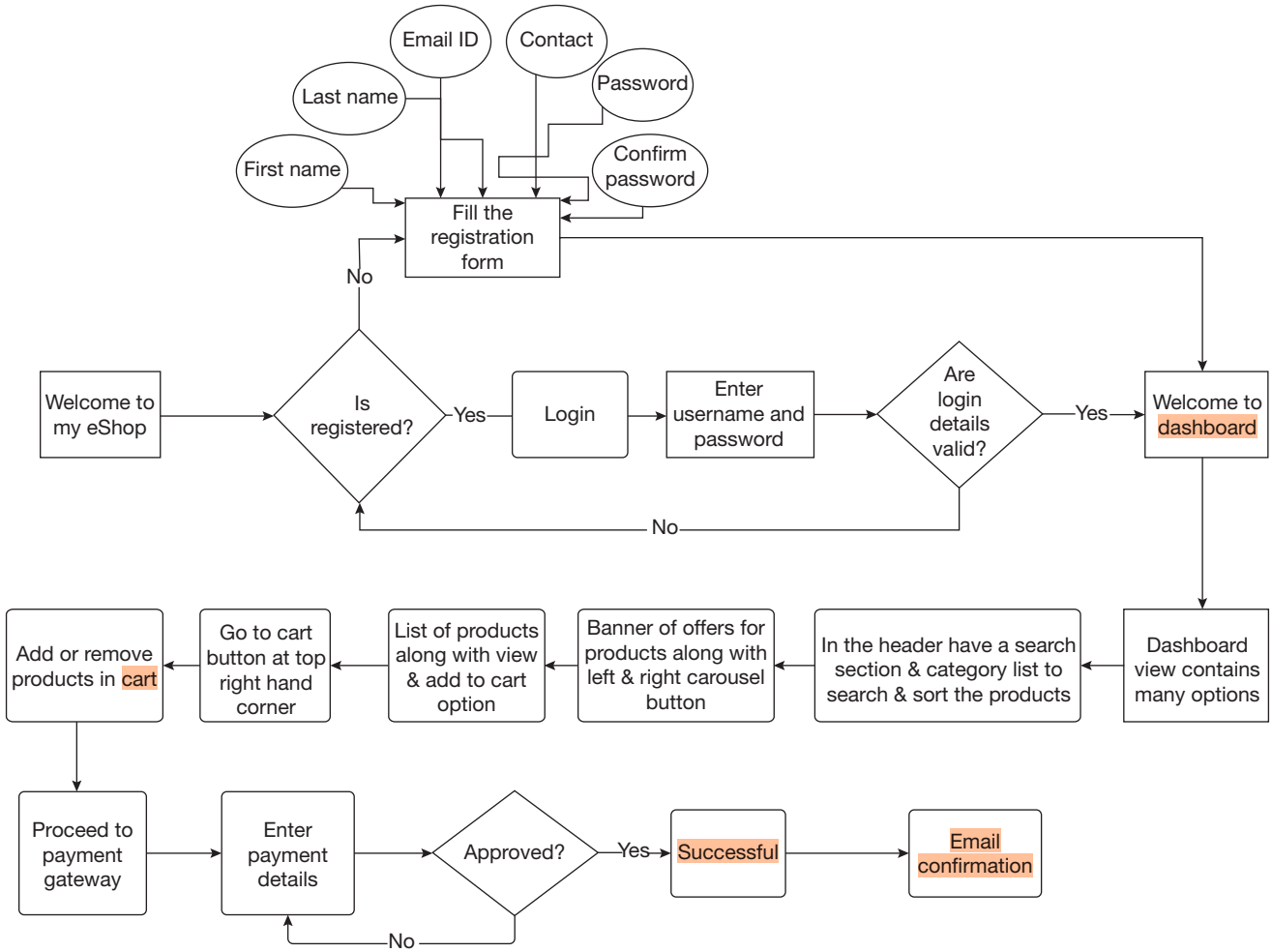


Figure 2.8 Login, registration, search, purchase, and payment flowchart.

QUICK CHALLENGE Expand this page flow diagram and add the detailed page elements to cover all the possible options and conditions.

2.9 | Back-End Web Services API Endpoints

We need to think of the endpoints that will be needed to get data from the back-end. Think of all the pages we need to feed data to and collect information from to store it in the database.

2.9.1 GET Endpoints

The GET method is used to transfer data from client to server. In this method, data is transferred via URL parameters and hence exposed to the world. Anyone listening to the traffic will see the data, as it is part of the URL. This method should not be used for login, registration, or any sensitive operations as it adds a huge security risk. Also, there is a limitation on sending data via this method as URL length is restricted. Following are some of the examples of GET endpoints.

```
"/customer/getCustomerByEmail/{email}"
```

The above endpoint allows us to fetch customer data using his/her e-mail address. The front-end collects customer e-mail and sends it to the back-end via GET method as mentioned in the endpoint by replacing {email} placeholder. The back-end then returns the customer information for that specific e-mail address in a JSON form. We can parse this JSON response on the front-end side to show the customer information. We have implemented this endpoint in the back-end for you to understand the code.

```
"/customer/getCustomerByID/{id}"
```

The above endpoint allows us to fetch customer data using his/her id. This endpoint works similar to the previous one where we were fetching customer data using his/her e-mail address. There is a slight difference on the back-end though where we will be using "long" type to accept "id" in the parameter list for the controller method compared to "String" in the record fetch via e-mail case.

```
"/order/cancel/{ordernumber}"
```

This endpoint is useful for cancelling the order. Since we need to pass order number only, we can use GET method for this.

```
"/order/status/{ordernumber}"
```

This endpoint is useful to check the status of the order. In this case also we only need the order number and hence GET can be used here.

QUICK CHALLENGE Based on the ERD, flowcharts and page flow diagrams that we have studied in this chapter, describe other GET endpoints you will need.

2.9.2 POST Endpoints

Following examples show us the use of POST in fetching data from the server. This method is very different from GET as it does not pass data via URL parameters. This method transfers information via HTTP header. The other advantage of POST over GET is that it does not have any limitation on data size so it can be used to transfer large photos, large texts, etc.

```
"/customer/save"
```

This endpoint is very straightforward from the URL point of view as the data is not getting sent via URL. This endpoint is there to create new customer data in the database. All the customer related information is required to pass via header so backend can access it in the same manner.

```
"/customer/update"
```

This endpoint as name suggests is available to update customer information in the database.

```
"/order/new"
```

This endpoint is useful to create a new order. Since order data may be sensitive in nature and moreover may contain large information, it is a good idea to send it via POST.

QUICK CHALLENGE Based on the ERD, flowcharts, and page flow diagrams that we have studied in this chapter, describe other POST endpoints you will need.

2.9.3 GET versus POST

In this section we will discuss the differences between GET and POST methods. As you have learnt, both methods are used to send data from client to server. However, they have a different working pattern, as listed in Table 2.1.

Table 2.1 Differences between GET and POST methods

	GET	POST
History	Parameters are passed as part of the URL and hence are stored in browser history.	Parameters are passed via HTTP header and hence do not get saved in browser history.
Bookmarked	As it is part of URL, it can be bookmarked.	Parameters cannot be bookmarked as header information does not get bookmarked.
Cached	GET is cacheable.	POST is not cacheable.
Re-submit/ Re-execute behavior	In case GET is used in the HTML form and gets cached, it cannot be resubmitted but it can be re-executed.	POST can be re-submitted but browser will alert user about it.
Encoding Type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data
Parameters limitation	URL length has some limitation on sending data.	No restrictions on data size.
Restrictions on form data type	In this mode, only ASCII characters are allowed.	There is no restriction on this and any type of data can be transmitted.
Hack severity level	Visible to public so it is less secure.	Data is sent via HTTP headers and in case of SSL, data is encrypted and hence not easy to hack.
Security	Since data can be stored in the history or bookmarked, it is available in a plain text and may expose to outside world.	This is not the case with POST so the data is not available to the outside world, making it more secure than GET.
Usability	Useful for only handful of small operations like fetching a record with ID or other parameters, etc.	Useful for sending any type of data including username/password, bank information, etc.

Flash Quiz Describe at least 10 advanced endpoints like predictive analysis for product selection, behavior tracking for buying choices, buying pattern for offers and promotions, etc.

There are many ways through which we can send data from client to server. However, only GET and POST are popular. The other methods are as follows:

1. **PUT:** This method is similar to POST method, but one major difference is PUT requests are idempotent which means calling the same PUT request will produce the same result. However POST will post data again and again, forcing the back-end to create multiple records.

2. **HEAD:** This method is similar to GET but one difference is it does have response body. For example, if GET method is fired on the customer controller with a parameter such as ID, server will return the customer data for that ID. However, in the case of HEAD, the data will not get returned as the response body is missing.
3. **DELETE:** This method is useful to delete a specified resource.
4. **PATCH:** This method is useful for the partial modification to a resource and does not allow complete replacement. This method is not idempotent like POST, so it will affect the data.
5. **OPTIONS:** This method describes the communication options for the resource.

Summary

As we have seen, application development takes a lot more effort than simply writing the code in a favorite language. There are various steps involved in designing and developing an application. Planning phase is equally important as it gives a clear blueprint of what needs to be developed. In this chapter, we have walked through the entire process and are now in a position to start working on our application. The next step is to gain the skills required to start the development process and develop further in the upcoming chapters. In this chapter, we have learned the following concepts:

1. Basic project idea, complexity, and implementation strategy.
2. Various architectural diagrams like flowchart, page flow, entity relationship diagram, etc.
3. Design thinking process and identify entities and classes.
4. Flow and relationships between entities and classes.
5. Web services API endpoints and their uses.

In the Chapter 3, we will learn about HTML, which is dominantly used in the industry to create front-end web pages. We will explore the basic building blocks of HTML and various tags, elements, and attributes. We will see some practical examples which will give you a clear idea on using those in a real-life project.

Multiple-Choice Questions

1. While creating Entity Relationship diagram, what is denoted by double ovals?
   - **(a)** Multi-value key
   - **(b)** Multi-value entity
   - **(c)** Multi-value table
   - **(d)** Multi-value attributes
2. Which of the following facilitates graphical representation of a given problem?
   - **(a)** Pseudocode
   - **(b)** Algorithm
   - **(c)** Flowchart
   - **(d)** All of the above
3. Which one of the following symbols is utilized at the beginning of a flowchart?
   - **(a)** Rectangle
   - **(b)** Circle
   - **(c)** Diamond
   - **(d)** None of the above
4. What is the main benefit of a User Flow Diagram?
   - **(a)** Spot potential problems
   - **(b)** Remove virus
   - **(c)** Draw comparison between two things
   - **(d)** Hide files
5. Derived attributes are denoted by _____ in entity relationship diagrams.
   - **(a)** Dotted Square
   - **(b)** Dotted Rectangle
   - **(c)** Dotted Triangle
   - **(d)** Dotted Oval

Review Questions

1. What is flowchart?
2. What is UML Class Diagram and what is its use?
3. In how many ways are entities are related to each other?
4. Why is entity relationship diagram essential?
5. How can one identify dependencies between entities?
6. What is the benefit of using POST method over GET?
7. When will you use PUT over POST method?
8. What is PATCH and when should one use it?
9. What is the benefit of using DELETE method over POST to delete a record?

Exercises

1. Create a flowchart for booking an airline ticket.
2. Define entities for airline booking web application.
3. Decide on classes that you need based on the entities and create a class diagram.
4. Draw page diagrams for the complete search and booking process.
5. Create a detailed comparison chart for POST, GET, PUT, HEAD, DELETE, PATCH, and OPTIONS.
6. Explain the benefits of architectural diagrams.

Project Idea

Study the following requirements of a taxi-booking web application and design a complete architectural diagram for the same. Explain all the entities and draw an entity relationship diagram. Based on these entities draw class diagrams and identify the relationships between these classes. Define a flow of the application and design page flow diagrams. Identify web service APIs that front-end will be calling and come up with data transfer methods for the API methods.

Taxi Booking Application: In an urban setup, it is extremely difficult to book a taxi as the only option for an individual is to wait for a passing by taxi and check if it is empty or not. In many cases, empty taxis are not available and the user ends up being canceling his trip or uses other mode of transport or simply walks if the destination is nearby. This problem also affects taxi drivers, as they do not get enough commuters to make a sustainable living. Having a taxi booking application where user can look for nearby taxies that are available and book one of them can solve this problem.

Recommended Readings

1. Axel Van Lamsweerde. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley: New Jersey
2. Hans Van Vliet. 2010. *Software Engineering: Principles and Practice, Third edition*. Wiley: New Jersey
3. Rajib Mall. 2018. *Fundamentals of Software Engineering, Fifth edition*. PHI Learning: New Delhi
4. Pankaj Jalote. 2010. *Pankaj Jalote's Software Engineering: A Precise Approach*. Wiley: New Delhi
5. Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. 2013. *Database System Concepts, Sixth edition*. McGraw Hill Education: New York